Fraunhofer
ITWM

CoPool

Pre-processor's manual

A. Zemitis,
O. Iliev,
T. Gornak,
B. Schmidtmann

CoPool revision 2.5.0
2/23/12
@Fraunhofer ITWM

Contents

1 Introduction

As described in [1], CoPool is a software tool used to simulate transient flow and heat transfer in containment pools. In order to perform simulations with CoPool, the user first has to provide geometrical information. These information are prepared with the help of the pre-processor CoPrep. The typical geometry of containment pool consists of wall surrounding the pool, walls separating the pool into connected or disconnected rooms, and various other solid or metallic walls, called here sometimes obstacles. Fig. 1 shows a cross section of a geometry which represents most of the peculiarity of a real containment pool.

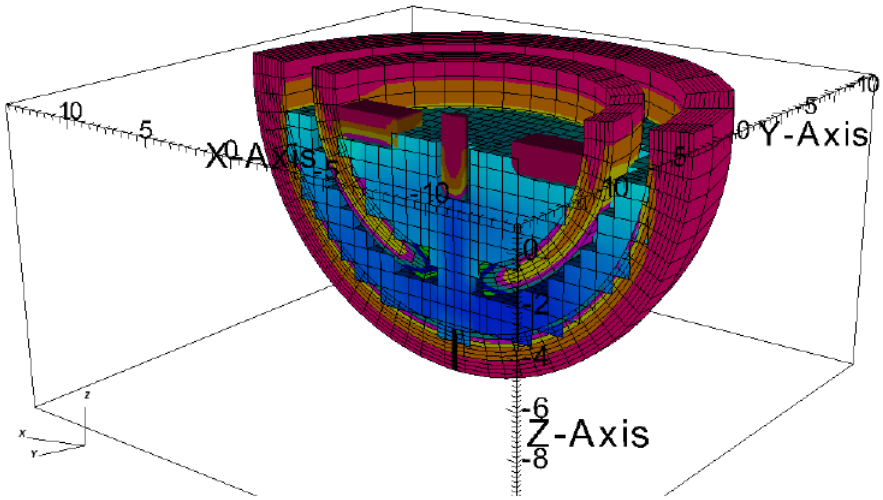

Figure 1 Typical pool geometry

As it can be seen from Fig. 1, the geometry can be rather complicated. The pre-processor CoPrep is developed with the idea that the creation of the geometry should be simple, as well as the generation of the grid for the computations, and the generated grid should provide pre-conditions for fast and reliable computations. To achieve this, the containment pool geometry is created from primitive geometry objects (plate, cylinder, sphere) and Boolean operations with them. The geometry shown on Fig. 1, for example, is created using hemispheres and plates, and a cylinder in the middle. The fluid occupies the region between the walls.

The construction of the geometry is done in two steps:

- the construction of the "exact" geometry using appropriate primitives

(cuboid, sphere, cylinder) and Boolean operations applied to these primitives,
- the discretization of the geometrical objects in an appropriate way.

The construction of a geometry always starts with constructing walls and/or other obstacles. As a post-processing of the created walls, the fluid domain is obtained. The discretization of the walls' geometry is then obtained as a collection of different local grids (one for each compound). Since the obstacles are build by the use of different primitives, which are discretized in appropriate coordinate systems, there is no global mesh for the obstacles. Each compound has a proper coordinate system which can be chosen and adapted by the user. For the description of the fluid grid, a Cartesian coordinate system is used. One can clearly distinguish different grids on Fig. 1. The main difference between the discretized fluid domain and the discretized walls' geometry is that the fluid part has a global mesh while all compounds that build the walls' geometry have local meshes. The global mesh of the fluid part covers different rooms, even when they are fully separated. Once the pre-processor has created all necessary information, then the CoPool simulation can be started. CoPool is capable of:
- simulation of fluid flow and heat transport in the fluid,
- simulation of heat transport in walls.

This manual describes how to generate geometric information using the pre-processor. The simulation parameters and simulation procedure are described in [1]. The examples mentioned in this manual can be found in directory Examples\CoPrep. Separate directory is created for each of the examples described here, and the name of the directory corresponds to the name of the example. In all cases the xml-file describing geometry is called Geometry.xml. Note: The examples used in this manual sometimes do not contain flow configuration part, and therefore can not be used for performing flow simulations. They illustrate the creation of the geometry and the generation of the grids in the walls. Examples prepared for performing flow simulations can be found in directory Examples/CoPool. The aim of this manual is to provide building blocks for users for constructing and discretizing the fluid part and the all parts.

This document is organized as follows. Next section explains how to run the pre-processor. Further on, section 3 discusses the basic concepts of the pre-processor, including definitions, compounds and Boolean operations, fluid grid and wall grids, as well as layers. Section ?? explains how to set the pre-processor parameters in order to generate adequate grid in the flow domain. The next three sections, Sec.5, Sec.6 and Sec.7, explain how to crate primitives (basic wall objects) how to transform them and how to perform Boolean operations with them in order to create a computational domain. One more section, Sec.9, deals with construction of wall geometry, this time in connection with treatment of the overlapping parts. Next, Sec.8 discusses the important questions of rooms' classification, which is crucial for performing flow simulations in case of flooding of many rooms. Finally, some concluding

remarks are drawn at the end.

2 Running the pre-processor

The pre-processor is build in the CoPool software. The full work flow of CoPool is described in [1]. The "Geometry.xml" file has to be stored in some project directory. The following steps has to be performed:

- Start the CoPool by clicking the short cut to CoPool software,
- Push the button "Open" and chose the project directory with necessary "Geometry.xml" file,
- Push the button "Preproc".

The pre-processor starts to work and produces necessary information about discretized geometry which is needed for performing simulations with CoPool. It is important to remember that pre-processor requires only correct information in the section <FileInput> of the "Geometry.xml". The second part <Document> of "Geometry.xml" is not needed if only pre-processor will be started.

3 Basic concepts

In this section we will give a terminology that uses in CoPrep, explain basic principles of operation. Reading this section is obligatory to understanding the manual.

3.1 Definitions

3.1.1 Input file

Before running the pre-processor, the user should provide an input file containing the geometric information of the problem. For this program, this file has always to be named Geometry.xml. The format of the input file is XML.

Using this link the user can download the xml editor Microsoft Xml NotePad.

The structure of XML documents can be seen in figure 2.
Each XML document starts with a root element. This element is "the parent" of all other elements. The terms parent, child, and sibling are used to describe relationships between elements. Parent elements have children. Children on the same level are called siblings. All child elements can again have sub-elements

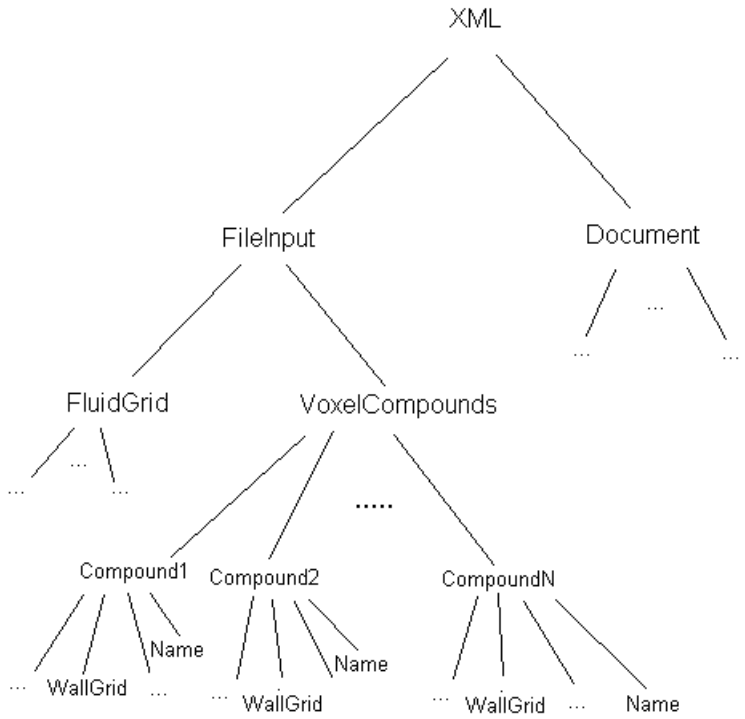

Figure 2 Outline of the hierarchy of an XML document.

In our case, the name of the root element is not important (you can simply call it XML, for example). The root element of the file Geometry.xml should have two child elements: FileInput and Document. In the section FileInput, the geometry with all its attributes is described, which is important for the

pre-processor. In the section Document, data for the simulation process is provided. It will be discussed in detail in the CoPool User's Manual. FileInput has again two child elements: FluidGrid and VoxelCompound that will be regarded in section 3.2.

3.1.2 Colors

Every geometry consists from voxels. The color of voxel depends on the type of object that includes this voxel. There are four main categories of colors:

- Rooms: color for room should always be between 2 and 100 (more about rooms one can find in a section 8).
- Bounding box: color of bounding box is 110. Bounding box is the solid layer that surrounds the fluid grid (for more detail see section 3.1.4).
- Walls: walls have color large than 110.
- Ovelapp: color large than 210 (see section 9 for details).

Colors are assigned to voxels automatically during geometry processing. After running CoPrep it should be checked that colors correspond to the correct type of objects.

3.1.3 Output files

The pre-processor creates the following files:

- vtk - files for the fluid part and all wall objects
- LeS - files for the fluid part and all wall objects
- Step size files for the fluid part and all wall objects
- SubDomainFusion.xml - containing room classification, air temperature, liquid temperature
- WallColors.xml - containing information about the colors used for walls.

vtk-files are very useful to illustrate the resulting geometry. They can be opened using special software as Paraview or Visit.
CoPrep generates a vtk-file for each named compound (if a compound has no name, there will be no vtk-file for it). Also, there will be a vtk-file for the fluid, therefore, the fluid should have a name that is set in the section FluidGrid.

LeS files are needed for simulations with CoPool.

3.1.4 FluidGrid

The FluidGrid describes the region and the mesh of the fluid. It can be filled with liquid but also with air. The FluidGrid is a cuboid, described in

rectangular coordinates, where the dimension and the location in space can be chosen by the user. The described cuboid will be surrounded by walls called bounding box and having the color 110. More information concerning the fluid grid will be given in section 3.3 and 4.

3.1.5 Elements (primitives)

The basic geometrical objects used for the pre-processor are primitives (cuboid, sphere and cylinder). Each primitive can be described by appropriate parameters as length, radius, etc. This information should be included in Geometry.xml. The user can construct all other objects on the basis of these three primitives.

3.1.6 Units

Units describes the unit corresponding to the grids. Possible values are m, cm and mm.

3.1.7 CoordSys

CoordSys determines the type of coordinate system used in the respective component. Possible values are rectangular, spherical and cylindrical. The three types will be discussed in the examples cuboid, sphere and cylinder, respectively (cf. sections 5.1, 5.2 and 5.3).

3.1.8 WallGrid

The section WallGrid describes the mesh of an primitive and is included in each compound. WallGrid is a sub-element of VoxelCompound and will be discussed in section 3.3 and in several examples, see section 5.

3.2 Compounds and Boolean operations

The element VoxelCompound has child elements called CompoundN, where N goes from 1 to the maximal number of compounds.
A compound is an object consisting of at least one primitive. In a more general case, the compound consists of several primitives connected by Boolean operations: conjunction, disjunction and negation. This means that compounds can be created by adding, cutting and overlapping different elements. The user should take care that the resulting geometry is still placed inside the initially defined fluid grid.

For each compound, the user should fill in the Name variable (for example “cylinder“) and the Color variable (an integer number). Also, it is possible to assign some material properties to the compound as InitialTemperature or MaterialType. If two elements shall have different material properties, they have to be built in separate compounds.

3.3 FluidGrid and WallGrid

In figure 3, a fluid grid section can be seen. In this example, the bounding box measures 300 x 300 x 200 and is centered around 0.

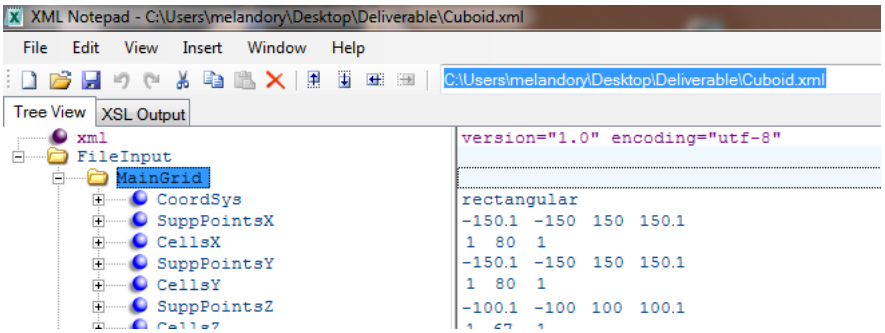

Figure 3 Parameters for a fluid grid.

Since the fluid grid always describes a cube, it has to be described in rectangular coordinates.

3.3.1 SuppPoints

This data field allows the user to choose support points for the x-, y- and z-direction (SuppPointsX, SuppPointsY and SuppPointsZ, respectively) (see figure 5).

Example in the figure 4 yields two intervals in x-direction: [-5, 5] and [5, 10].

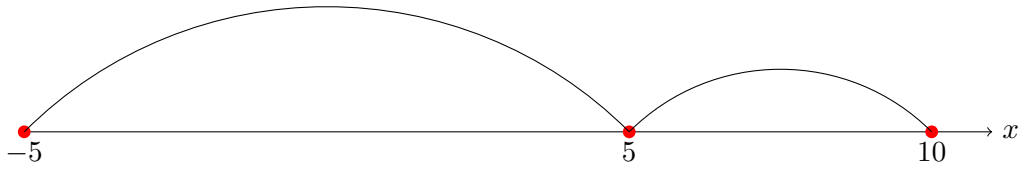

Figure 4 SuppPoints: intervals [-5, 5] and [5, 10]

3.3.2 Cells

Cells allow to choose the number of cells in each interval. Here again, this choice has to be done for the x-, y- and z-direction (called CellsX, CellsY and CellsZ, respectively).
For the two intervals in example in the figure 4, two values have to be set (one for each interval). In the figure 5 is shown how it should be done.

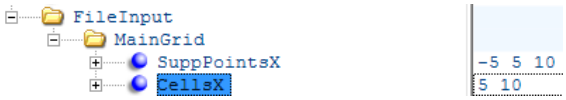

Figure 5 Parameters for number of cells in each sub-interval (in this case 5 and 10).

We show in the figure 6 how cells definition works (cells in the first interval and 10 cells in the second one).

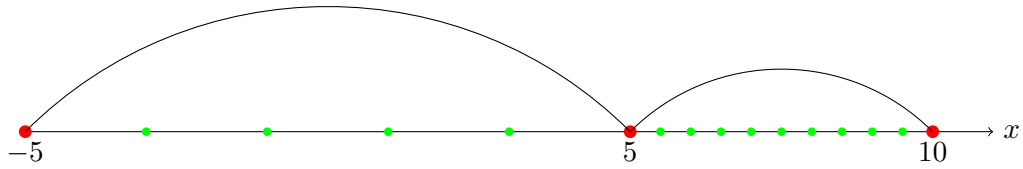

Figure 6 SuppPoints: 5 points in interval [-5, 5] and 10 point in interval [5, 10]

In the WallGrid of each compound, one can choose between rectangular, cylindrical and spherical coordinates and therefore, suitable support points are necessary: For a cylindrical grid the user should set SuppPointsR, SuppPointsT and SuppPointsZ and corresponding cells (CellsR, CellsT, CellsZ). For a spherical grid, one should set SuppPointsR, SuppPointsT and SuppPointsP and the corresponding cells (CellsR, CellsT, CellsP).

3.3.3 Placing objects

If the desired object is a circle with center (0, 0) and radius 2 (see figure 7).

The minimum x-value of object in the figure 7 is -2 and its maximal x-value is 2. It is important that the minimal value of SuppPointX is less than -2 and the maximal value greater than 2. For example, SuppPointX $= -3, 3$ is a possible interval (shown in the figure 8 with CellsX $= 12$).

Also, SuppPointX $= -2.1, -2, 2, 2.1$ with CellsX $= 1, 8, 1$ is a valid choice. In figure 3, the support points -150.1 and 150.1 have been added in x-direction. For each of these two extra points, an extra cells has be created, too.

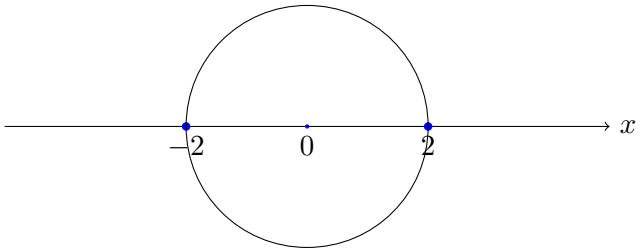

Figure 7 Circle

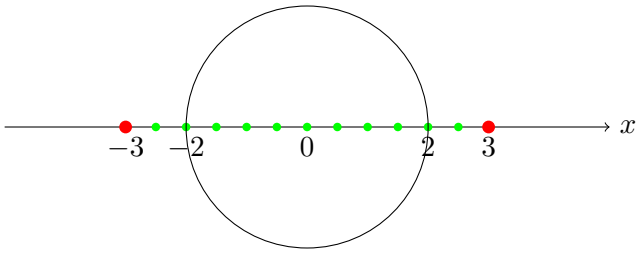

Figure 8 Circle with cells

3.4 Layers

Possible values of Layers are 1 and -1:

- 1 means that the object is simply placed in the FluidGrid.
- -1 means that the object is cut out. If object_1 has layers = -1 and it overlaps with object_2 which has layers = 1, then object_1 is subtracted from object_2. The subtracted area is empty, which means that it can be filled with fluid.
- Layers does not necessarily need to be set. In case it does not exist in a compound, it is considered as 1 (see figure 9).

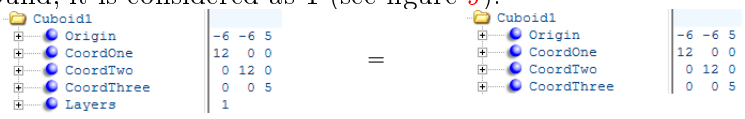

Figure 9 Cuboid with and without data field Layers.

4 Discretization of the fluid domain

First of all, the user has to decide about the size of the domain that he is interested in. Also, it should be thought of the different possibilities to position the domain - it can be centered around the origin, one can only consider the positive semi-axes, etc.
Here, a centered fluid grid is chosen. In all three directions, it goes from -20 to 20.
Remark:
Keep in mind that the fluid domain and the fluid grid has to be sufficiently large, such that all compounds fit completely into the fluid domain!
Once the size of the bounding box has been fixed, the user has to decide about the number of voxels in each direction. Also, some support points can be fixed in space.
For any fluid domain, it is important to add at least one boundary cell on each side. In the figure 10 extra fluid cells describe the bounding box of the fluid domain. For the example shown on figure 10, since the domain goes from -20 to 20, the support points are chosen at -20 and 20 but also at -20.5 and 20.5 to construct some extra cells. In total, five extra cells on each side have been added in this example.

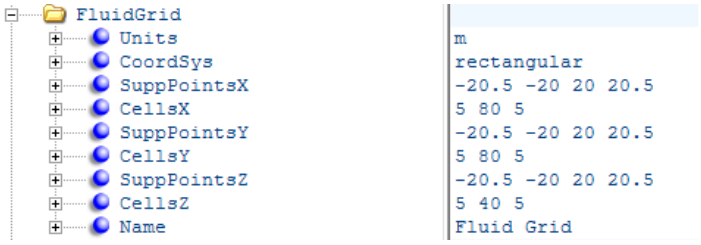

Figure 10 Parameters for of the fluid grid.

If an obstacle is positioned on the boundary of the fluid grid and there is no extra cell, the simulation will not work correctly because the obstacle is not completely covered with fluid. In figure 11, an example of such a case where a cylinder sticks out of the fluid domain (because it is placed at the boundary and no extra cell exists) can be seen.

Until now, the support points have only been set at the boundary and outside of the domain to create extra cells. Another possibility is to fix some points at boundaries of objects inside the fluid domain to force the program to match fluid voxel faces and faces of the obstacle. As an example, let us consider a cube with side length 5.4m. The cube will be centered at zero, i.e.
$x \in [-2.7; 2.7], y \in [-2.7; 2.7], z \in [-2.7; 2.7]$.

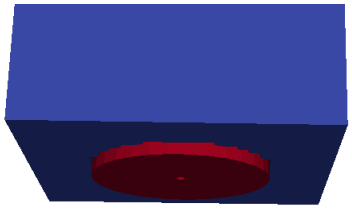

Figure 11 Cylinder that sticks out of the fluid domain.

If the fluid grid is fixed as seen above and the support points of the wall grid are chosen regarding only the wall itself, the following might happen (see figure 12). The exact wall size is larger as the wall size represented on the fluid mesh.

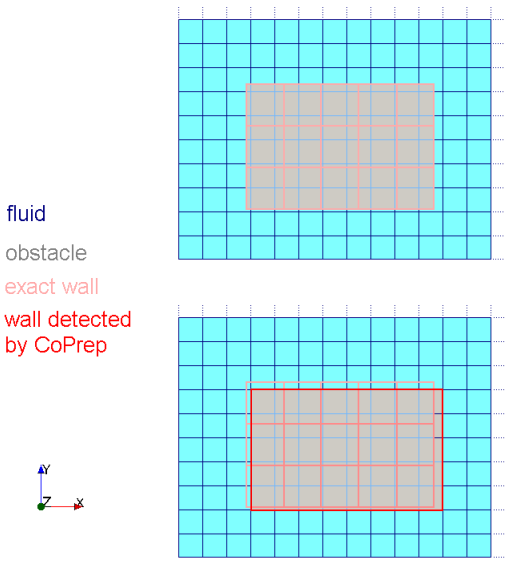

Figure 12 Differences in wall size representation in fluid mesh and the exact wall size.

It is therefore useful to fix the support points of the fluid grid at -20.5, -20, -2.7, 2.7, 20, and 20.5. In this case, one can be sure that the wall will be treated exactly as desired.
Remark: If more support points are chosen, an appropriate number of cells has to be chosen, too. Here, instead of 5 80 5 the user can for example choose 5 30 20 30 5. The corresponding mesh can be seen in figure 13.

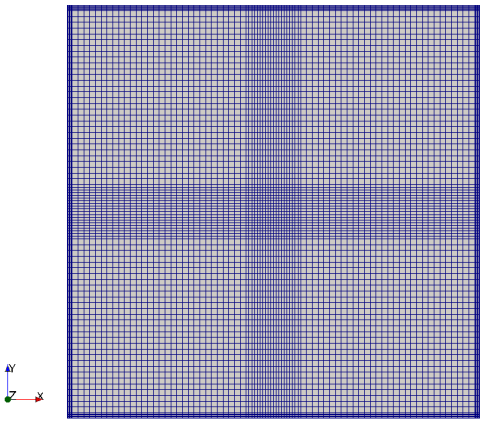

Figure 13 Different discretization for the boundary, the obstacle (in the middle) and the fluid part.

5 Primitives

In the following section we will describe and show with pictures how to create basic primitives. Previously, in section 3.2 it had been explained that one can construct geometry from primitives using boolean operators. After reading this section you will know what primitives you can create in CoPrep, what coordinate system does CoPrep use and how to operate with them, how to create input files and a little about post processing the result. While reading this section you can notice that there is no information about some parts of an input file (for example, Transformation), missing information will be explained in the next sections.

Examples from this section can be found in the folder Examples CoPrep. Each example is stored in a separate directory and the file name is Geometry.xml. The first examples are simple primitives as cuboid, sphere and cylinder. Later, in section 7, the interaction of different primitives will be considered. With the help of these examples, the user shall learn how to create geometries by himself and how to use the pre-processor.

5.1 Primitive named "Cuboid"

Open the file Cuboid\Geometry.xml with an Xml-editor. The file content is shown in figure 14.

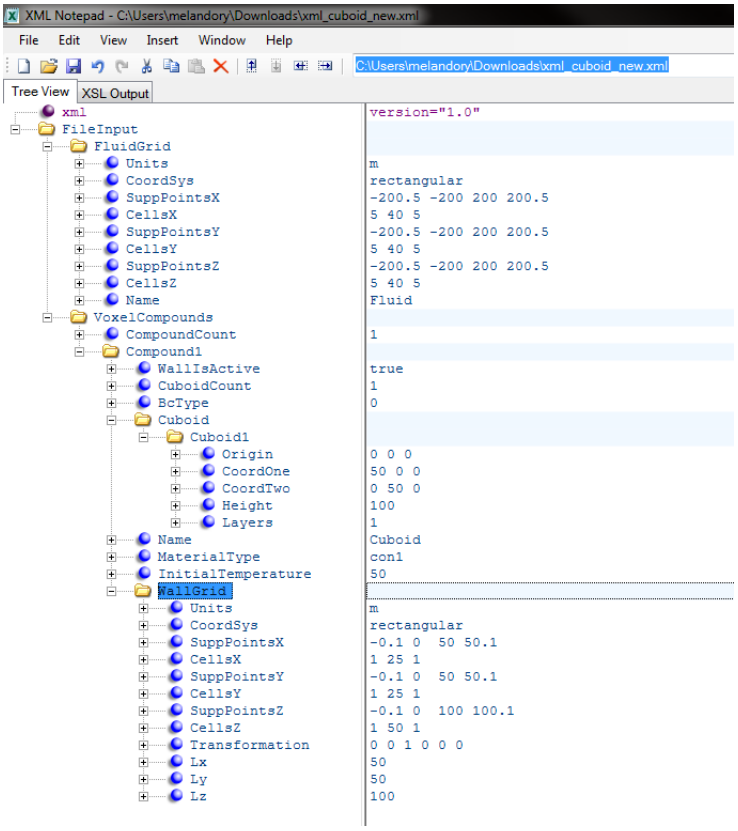

Figure 14 Parameters for the fluid mesh and the compound Cuboid

Now run the pre-processor as described in section 2. While the pre-processor is running, it displays on the screen information about all wall parts (primitives) of the geometry:

Name	WC	CS	MT	IT	BCC	CL
COPREP OUTPUT/Cuboid	111	0	Uninitialized	0	1	2

where

WC: Wall Color
CS: Coordinate System (0=rectangular, 1=cylindrical, 2=spherical)
MT: MaterialType
IT: Initial Temperature
BCC: Boundary Colour Count
CL: Color List

The information displayed above correspond to an example where neither the material type, nor the initial temperature have been initialized. Therefore,

InitialTemperature is zero (this is the default value) and the field MaterialType is marked as uninitialized. In this such a case, e.g., the example (CoPrep\Cuboid\), default values are used and these are as follows: the material type is con1 (this stands for concrete 1) and the initial temperature is 50.

When CoPrep has finished, a new folder named COPREP_OUTPUT has been created. There are important files in COPREP_OUTPUT that will be analyzed later on. For now, the interesting output file is the vtk-file Cuboid.vtk. This file serves to visualize the geometry, using for example Paraview.

Start paraview, then open the vtk-file generated by CoPrep. Remember that the variable Name of the only compound treated in this example has been set to Cuboid and that the fluid grid's name is Fluid. That is why we obtain two vtk-files: Cuboid.vtk and Fluid.vtk. In the Paraview-menu choose File → Open. Choose the folder COPREP_OUTPUT and then the file Fluid.vtk. In the pipeline browser on the left hand side, Fluid.vtk appears. Press the button Apply or use the shortcut Alt+A and the fluid domain appears in the main window.

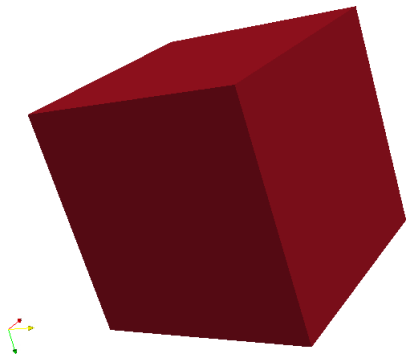

Figure 15 Bounding box of the fluid.

In figure 15, one can see the solid layer that surrounds the fluid grid. This layer has the color 110 and is the bounding box of the fluid.

As it was mentioned above, CoPrep displayes wall colors, and in the presented example the wall color of the cuboid is set to 110. In order to see the cuboid in the fluid grid, Paraview should visualize only the part of the geometry that has the color 110. To obtain this part of the geometry, use the tool Threshold. To create a threshold, use the path Filters → Recent → Threshold. We show in the figure 16 how to create threshold in Paraview.

In the Object Inspector on the lower left, set the value of Lower Threshold and Upper Threshold to 110 and press the Apply button. Eventually, the user

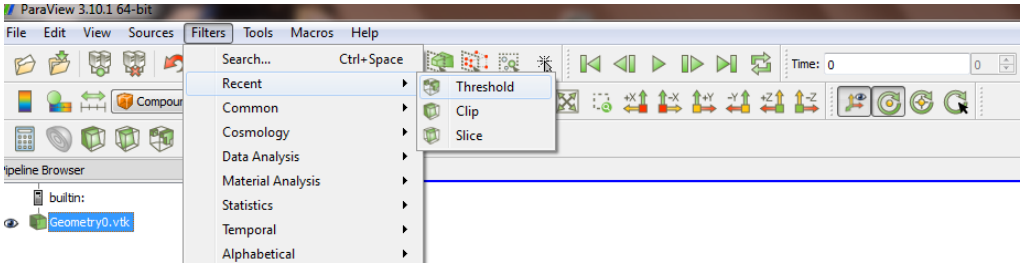

Figure 16 Applying threshold in ParaView.

has to activate the section Object Inspector by clicking on View and then checking the box 'Object Inspector' (see figure 17) .

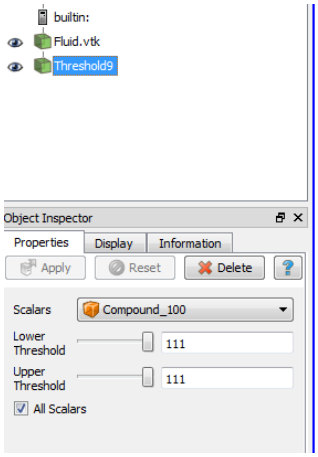

Figure 17 Object inspector.

The resulting geometry is shown in figure 18.

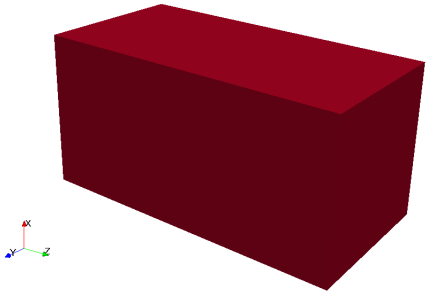

Figure 18 Visualization of the Cuboid.

Rectangular Coordinates

To describe a rectangular compound, the user only needs to specify the length in x- and y-direction and the obstacle's height. In the above example, a cuboid with dimensions 50m x 50m x 100m has been considered. Taking a closer look at the wall grid, one can observe that the support points for x (SuppPointsX) have been set at 0 and 50 (since the cuboid's origin is 0 0 0 and it measures 50m in x-direction). Also, there are support points at -0.1 and at 50.1 with one cell between -0.1 and 0 and one between 50 and 50.1. This serves to create an extra layer surrounding the cuboid. This additional layer is used for placing the information about the boundary conditions, as explained above in Sec.4. During the mesh generation, the thickness of this extra layer becomes zero. Therefore, its actual size does not play any role. In the same way, the support points and cells for y and z are constructed.
In this example, the data field Transformation does not have any effect on the cuboid, since the first three entries are left to 001, which means that the z-axis is the directional vector of the compound. Since the last three entries are set to 0, the cuboid's origin is not shifted in space. For more information concerning the transformation of compounds, cf. section 6.
The last three items of the wall grid are Lx, Ly and Lz. These are the characteristic lengths of the cuboid in x-, y- and z-direction which have to match with CoordOne, CoordTwo and Height. We display in the figure 19 the part of xml file with WallGrid section.

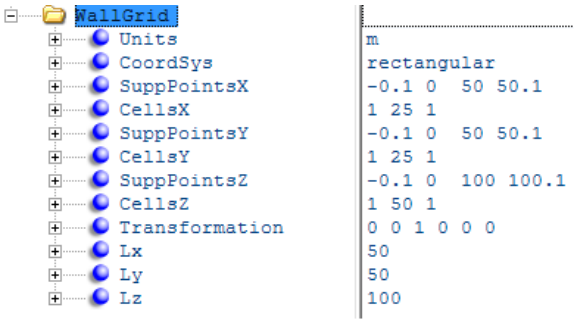

Figure 19 Example for a wall grid in rectangular coordinates.

5.2 Primitive named "Sphere"

To describe a sphere, there are two possibilities: one can describe the sphere in rectangular or in spherical coordinates. Both cases are described below, even though it is strongly recommended to describe spherical objects in spherical coordinates!
In both cases, the user needs to know the coordinates of the center of the sphere (Origin) and the desired radius (Radius). If the sphere is described in rectangular coordinates, these information are provided in the xml-file as can

be seen in Sphere\Geometry.xml. In the figure 20 corresponding xml file is shown.

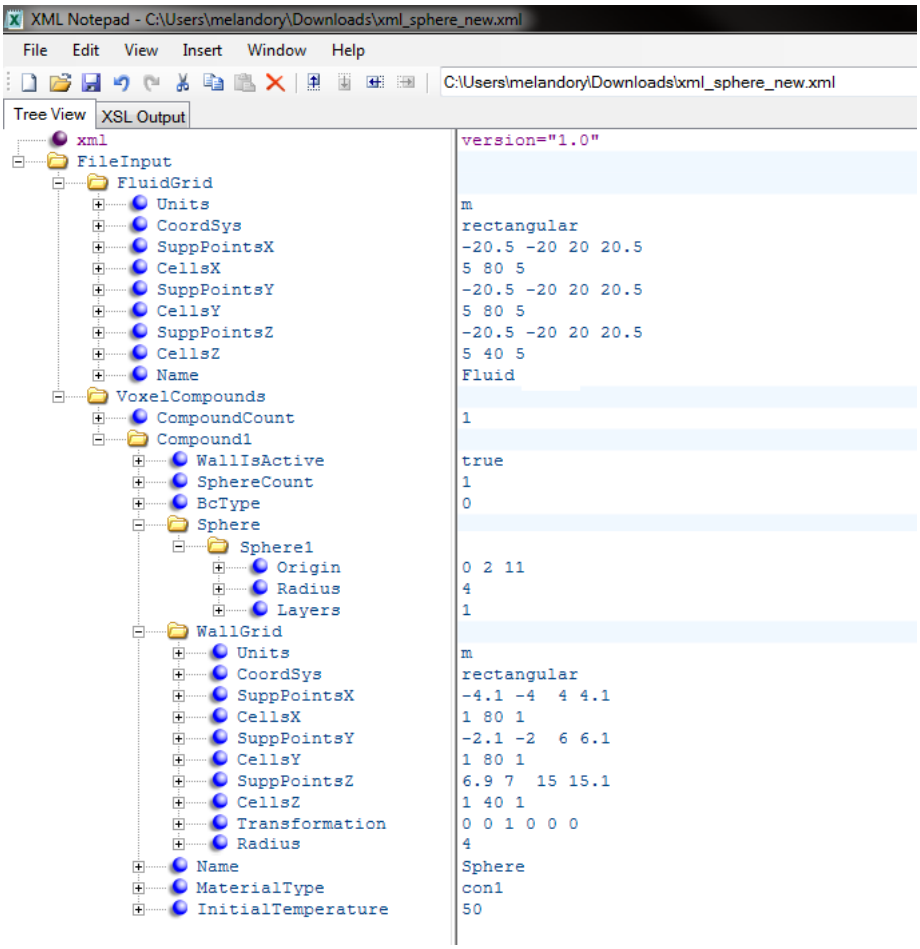

Figure 20 Parameters for the fluid mesh and the compound "Sphere".

The resulting geometry is shown in figure 21.

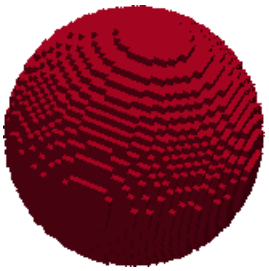

Figure 21 Visualization of sphere discretized in Cartesian coordinates.

Spherical Coordinates

To describe an obstacle in spherical coordinates, each point of the object is fixed by the radial distance from the origin, its inclination angle θ and the azimuth angle φ i.e. by R, θ φ, where $R \geq 0$, $\theta \in [0, \pi]$ and $\varphi \in [0, 2\pi)$.
Here, θ is abbreviated by T and φ by P.
We show in the figure 22 example of a Wall Grid in spherical coordinates (for a sphere with radius 4).

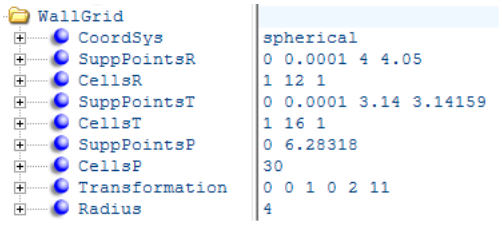

Figure 22 Parameters defining wall mesh in the case of spherical coordinates.

If the sphere is described in spherical coordinates, it is mandatory to set its Origin to $(0, 0, 0)$. Via the data field Transformation (child of wall grid), the sphere can then be shifted to the desired position. To obtain the same position of the origin as in the example above, namely $(0, 2, 11)$, the xml-file (Sphere_sphCoords\Geometry .xml) is shown in figure 23.

It is important to set radial support points not only at 0 and the radius but to include a point near 0 and to include another support point that is slightly larger than the radius. Also, for the inclination angle, it is important to add an extra support point with one or more cells inside the interval $[0, \pi]$.
The data field Transformation is left the same as in the above treated example. The last entry, Radius, describes the characteristic length of the sphere.

The first three entries of Transformation describe the orientation of the main axis in space. By default this is the z-axis, i.e. setting the first three entries to 0 0 1 or 0 0 0 does not change anything. For a sphere, this is not very important. Transformation is regarded in detail in section ?? and in section 6. The last three entries of Transformation describe the translation of the origin in space, therefore, in this example, Transformation is set to (0 0 1 0 2 11).

The resulting geometry can be seen in figure 24.

Remark:
One can observe that the sphere looks a lot smoother in spherical coordinates than in rectangular ones, even though the resolution was lower. Therefore, it is strongly recommended to describe spherical objects in spherical coordinates!

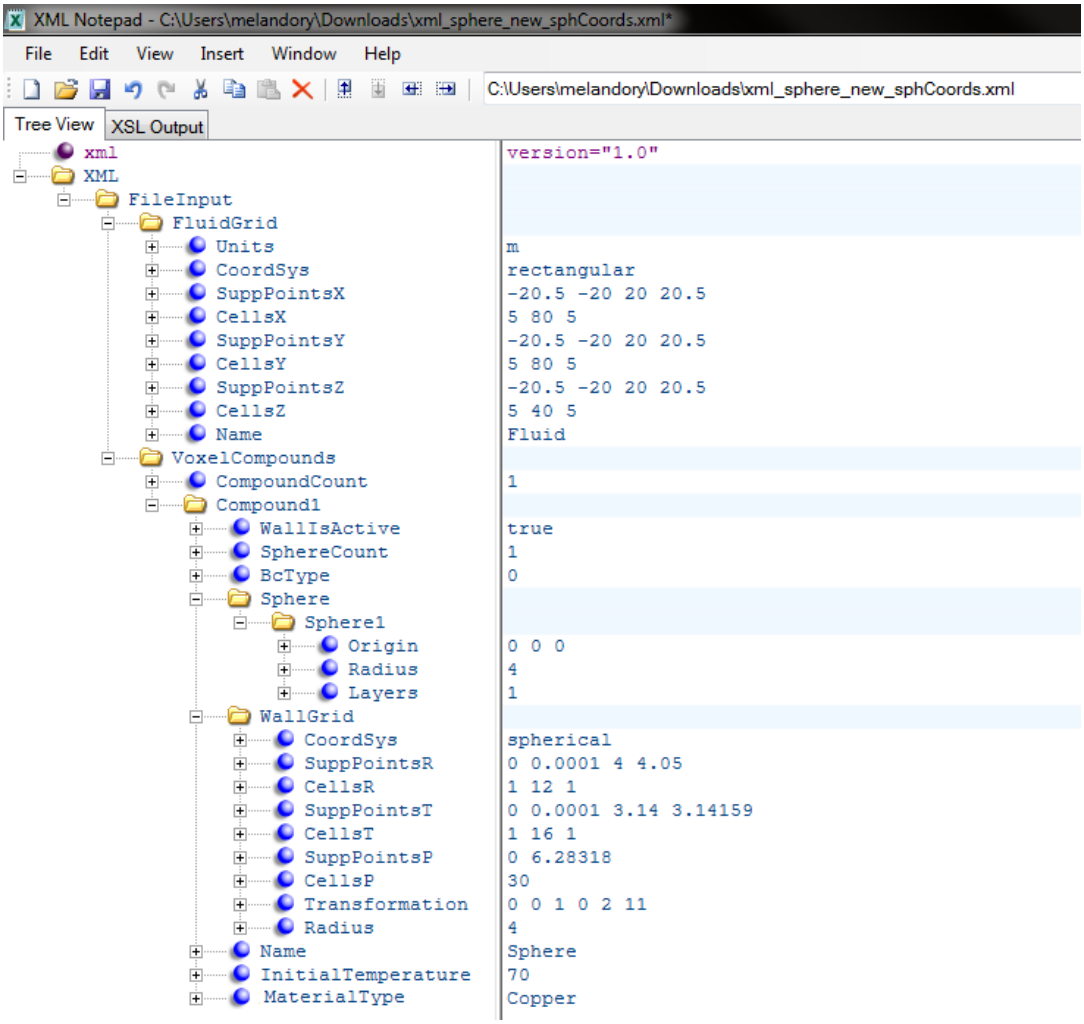

Figure 23 Fluid mesh and Sphere compound described in spherical coordinates.

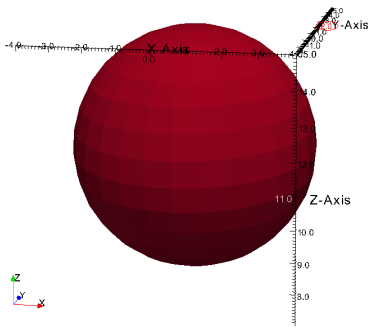

Figure 24 Visualization of sphere in the case of spherical coordinates.

5.3 Primitive named "Cylinder"

To describe a cylinder, there are again two possibilities: one can describe it in rectangular or in cylindrical coordinates. Again, it is strongly recommended to describe cylindrical objects in cylindrical coordinates!
In both cases, the user has to fix the cylinder's main axis as parallel to the Z-axis (CenterLine_P 0 0 1).
In general, the cylinder can either be parallel to the X-axis, parallel to the Y-axis, parallel to the Z-axis or even a combination, i.e. it can be arbitrarily positioned in space. For more detail see examples below and section 6. For the complete description of the cylinder, the coordinates of the origin (Origin), the radius (Radius) and its height (Height) have to be provided. In the figure 25 one can find the xml-file of a cylinder described in rectangular coordinates (Cylinder\Geometry.xml)

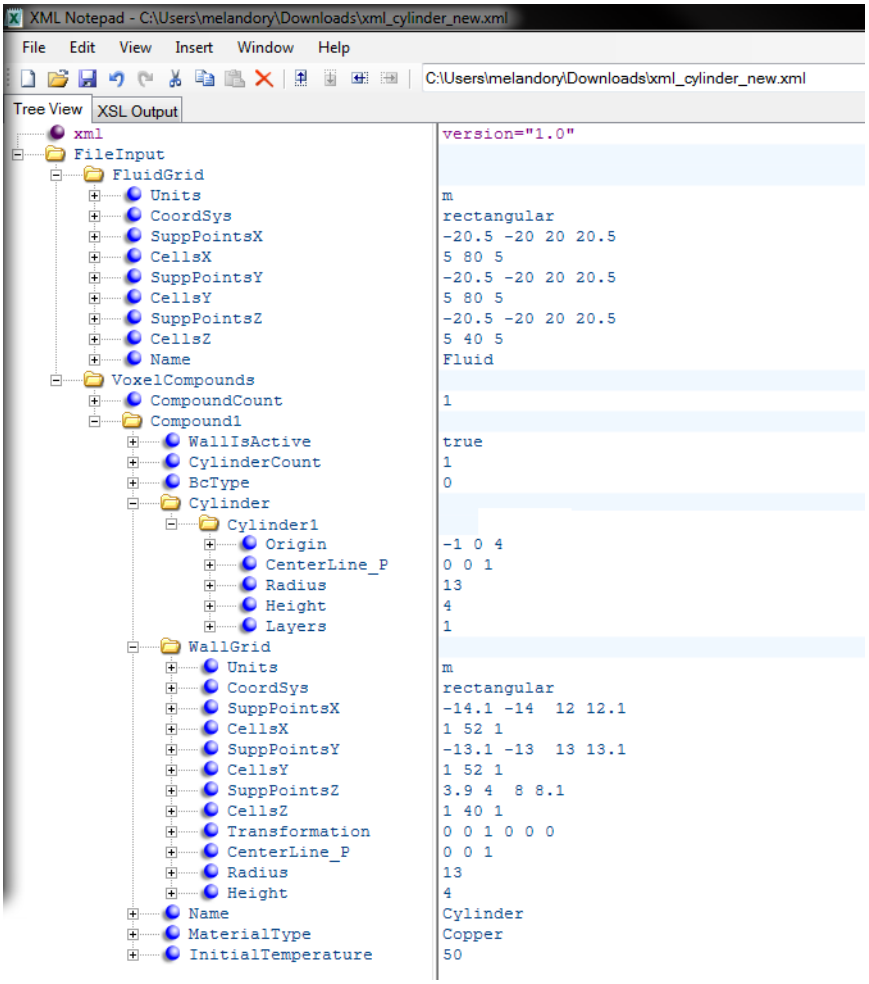

Figure 25 Parameters for the fluid mesh and the compound Cylinder in the case of Cartesian coordinates.

The transformed cylinder can be seen in figure 26.

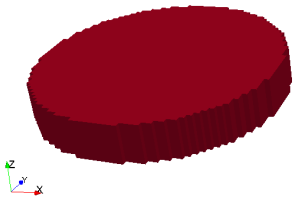

Figure 26 Cylinder in the case of Cartesian coordinates.

Cylindrical coordinates

To describe an object in cylindrical coordinates, one needs the radial distance from the origin R, the angular coordinate θ (here called T) and the height z, where $\theta \in [0, 2\pi)$. To describe the previous cylinder in cylindrical coordinates, it is important to set its Origin to $0, 0, 0$. The transformation of the origin will be affected via the last three entries of the data field Transformation, which is a child of WallGrid. The xml-file for a cylinder in cylindrical coordinates (Cylinder_cylCoords\Geometry.xml) is shown in 27.

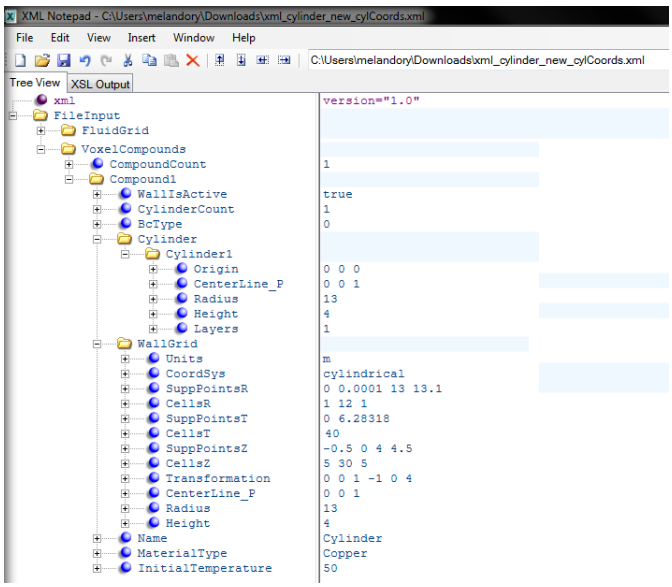

Figure 27 Parameters for the fluid mesh and the compound Cylinder in the case of cylindrical coordinates

In the figure 28 the wall grid of a cylinder with radius 13m and height 4m in cylindrical coordinates is depicted.

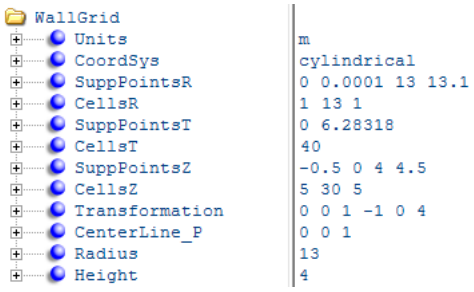

Figure 28 Mesh parameters for the cylinder in cylindrical coordinates.

It is important to set radial support points not only at 0 and 13 (the radius) but to include an extra support point near 0 and another one that is slightly larger than the radius (here 0.0001 and 13.1). Also for the height, the user needs to add at least one extra cell at each extremum. In this example, support points in z-direction have been set at 0 and 4, i.e. at the boundaries of the cylinder but also at -0.5 and at 4.5 (or at -0.1 and 4.1, etc.). Five cells have been assigned to both extra intervals.
The data field transformation is left as in the example above and has been explained in its context.
The last three entries, CenterLine_P, Radius and Height serve as characteristic lengths of the cylinder. They have to match with the center line, the radius and the height used for the description of the cylinder.

The tranformed cylinder in cylindrical coordinates is shown in figure 29.

One can clearly see, that the cylinder is a lot smoother when described in cylindrical coordinates. It is therefore strongly recommended to describe cylindrical objects in cylindrical coordinates!

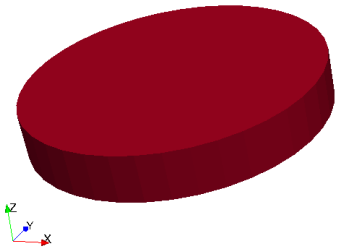

Figure 29 Cylinder in the case of cylindrical coordinates.

Visualization

We now want to take a closer look at the visualization of the the cylinder. Start Paraview and open the file Fluid.vtk as it has been described for the cuboid in section 5.1. Let us now use the Clip filter. To create a Clip, use the path
Filters → Recent → Clip. In the Object Inspector on the lower left hand side, the clip-settings panel can be seen (see figure 30).

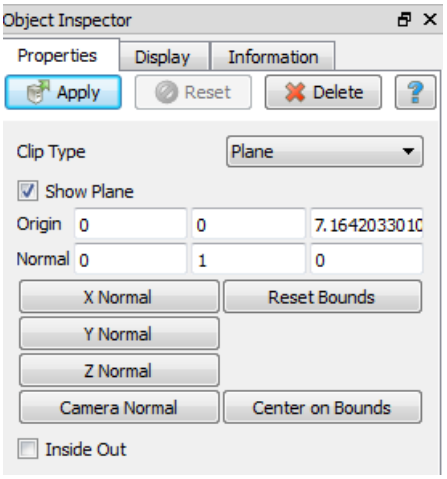

Figure 30 Object inspector in Paraview.

With this panel, the user can set the position and direction of the clip plane. One can also directly click on the plane in the main window and change its position by dragging it. In the figure 31, a clip in X-direction, positioned in the middle of the fluid grid can be seen.

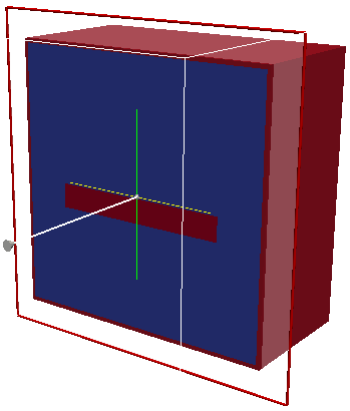

Figure 31 Clip operator in Paraview

One can see the bounding box (red wall surrounding the fluid grid). The blue

part is a fluid (having the dimensions of the fluid grid that has been defined by the user) and the red part in the center is the cylinder that was defined in the example above. Here, the cylinder is completely embedded in the fluid part. This should always be the case!

6 Transformation

Starting from the CoPool version 2.5.0 there is a new element in Geometry.xml available. The data field Transformation is a child of WallGrid and is necessary to shift and rotate obstacles in space. Here shortly the parameters are explained using a cylinder. Actually all wall objects can be transformed. If this data field is not included in the WallGrid then no additional transformation to this wall object is applied.

The data field has six entries.
The first three entries describe the orientation of the new z axis in space. The vector describing the new position of the z-axis does not need to be normalized (the norm of this vector does not need to be 1). By default, the z-axis will remain in the original position. If the user creates for example a cylinder and sets Transformation to $0\,0\,1\,0\,0\,0$ (or $0\,0\,c\,0\,0\,0$ where c is a number or simply $0\,0\,0\,0\,0\,0$) the rotational axis of the cylinder will be the z-axis. In the figure 32 this cylinder is shown.

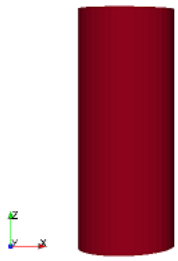

Figure 32 Cylinder by default Transformation = 0 0 0 0 0 0.

If Transformation is set to $1\,0\,0\,0\,0\,0$ (or $a\,0\,0\,0\,0\,0$ where a is a number) the rotational axis of the cylinder will be the x-axis. In the figure 33 the transformed cylinder is shown.

Figure 33 Cylinder transformed by Transformation = 1 0 0 0 0 0.

The last three entries of Transformation shift the obstacle in space. The fourth parameter indicates how far the obstacle is translated along the x-axis, the fifth parameter how far along the y-axis and the sixth parameter how far along the z-axis.
In figure 34, a cylinder with main axis in z-direction has been shifted 7m in x-direction, 2m in y-direction and 3m in z-direction.

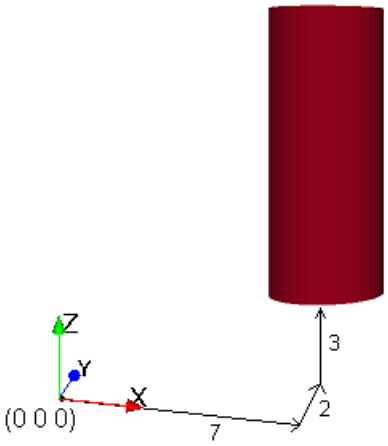

Figure 34 Shifted cylinder by using Transformation = 0 0 1 7 2 3.

Therefore, the data field Transformation reads 0 0 1 7 2 3 .

Transformation of the cylinder from section 5.3

In this section we will consider transformation of a cylinder from section 5.3.

The data field *CenterLine_P* has always to be fixed to 0 0 1. In case the user wants to orient the cylinder in any other way, the field Transformation has to be used:
(1) cylinder oriented along the x-axis (figure 35)
(2) cylinder oriented along the y-axis (figure 36)
(3) cylinder oriented in space (figure 37)

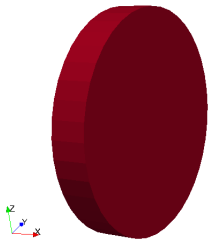

Figure 35 Transformation = (1 0 0 0 0 0)

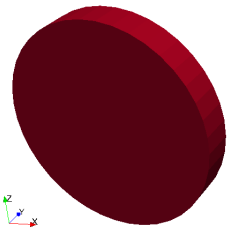

Figure 36 Transformation = (0 1 0 0 0 0)

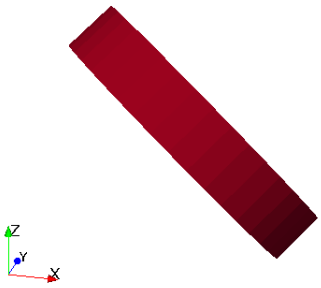

Figure 37 Transformation = (a b c 0 0 0)., where a, b, c are appropriate values, here a=b=c=1.

7 Working with primitives

In section 5 it had been listed what primitives you can create with CoPrep. But our goal is creating complex geometries. To obtain more complex geometries, different primitives are put together in the fluid grid. The examples treated in this section combine different types of primitives.

7.1 Pool

To start with a simple example, let us create a square pool. At first, the user should create a cuboid (see section 5.1). We show in the figure 38 parameters for Cuboid.

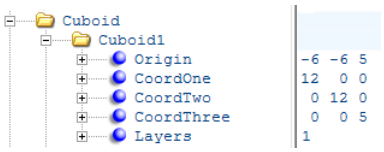

Figure 38 Parameters for a Cuboid.

Then, in the middle of the cuboid, in order to create a pool, some part has to be removed (this part can be of any form but for the sake of simplicity, we consider a second cuboid). To remove a cubic part, another (smaller) cuboid has to be described and its variable Layer has to be set to -1. This means that it is cut out. The second cuboid should be located inside the first one in order to create a pool (see figure 39).

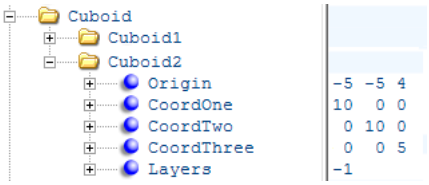

Figure 39 Parameters for a squared pool.

The resulting wall geometry can be seen in figure 40.

Figure 40 Visualization of the wall object.

The file for this example can be found in the folder Examples\CoPrep \Pool.

7.2 Cylindrical pool with obstacle

As a last example of this kind, a geometry that contains all types of elements is considered. First, a cylindrical pool is described. This pool is created in a

different way than the previous one (cuboid pool). The cylindrical pool will consist of three cylinders. One for the bottom and two cylinders to form the walls (see figure 41).

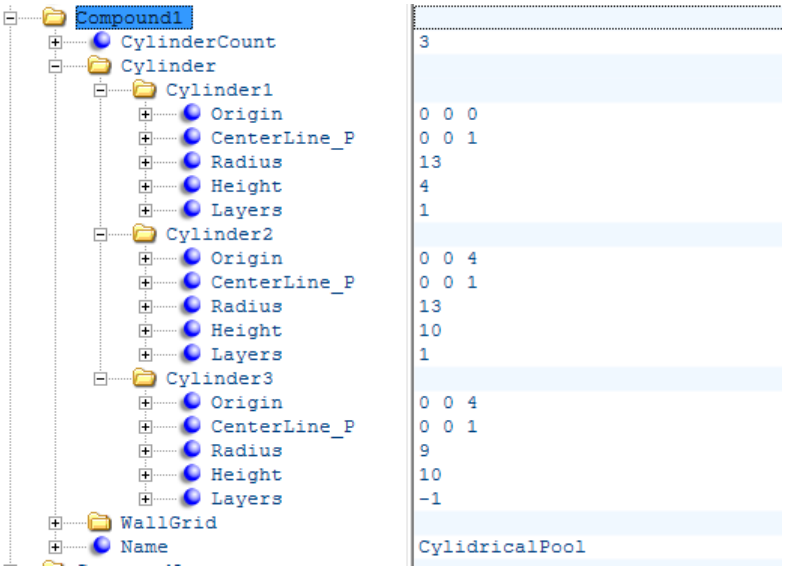

Figure 41 A compound is build using 3 cylinders.

The three cylinders are part of one single compound (Compound1), named CylindricalPool. To add another object, create a new compound (Compound2). In this example, we create a rectangular pool inside the cylindrical one and then add a hemisphere inside the rectangular pool. To do so, the spherical and cubic walls are described in two different compounds. In figure 42 the part of xml file for cubic walls us shown.

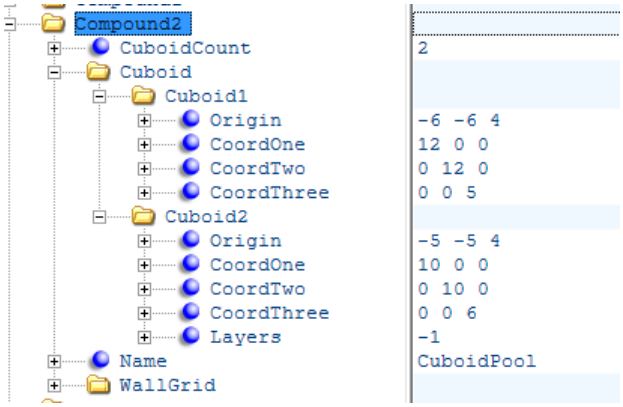

Figure 42 Parameters for the CuboidPool

To receive a hemisphere, one has to create a whole sphere, remove a smaller

sphere inside such that the sphere is hollow and then cut off one half by using a large cuboid which has the value Layers set to -1 (see figure 43).

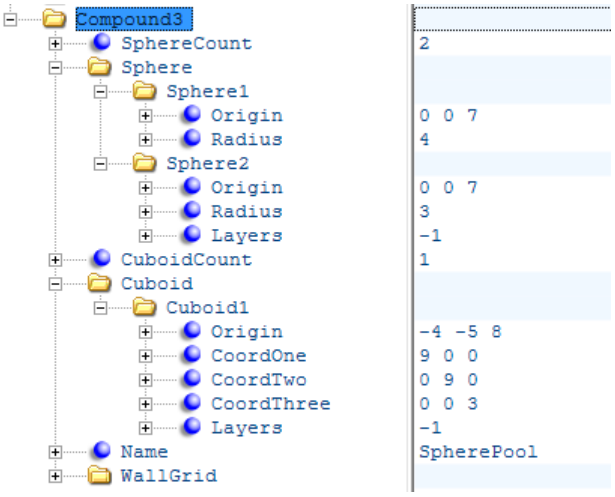

Figure 43 Parameters defining a Hemisphere.

When all three compounds are created, we can copy the xml-file in the right folder and run coprep.exe. We obtain the folder COPREP_OUTPUT, that contains the vtk-files that are needed for the visualization. Open the file Fluid.vtk with Paraview and create a Threshold (see section 5.1). In this example, set the value Lower threshold to 110 and the value Upper threshold to 112. The upper value has to be 112 since we have three compounds. The numbers of the colors of the compounds start with color 110 for Compound1. Compound2 has color 111 and Compound3 color112. In the left panel of figure 44 we display clip of Fluid.xml file and at the right panel of the figure 44 we display clip of xml files for different compounds.

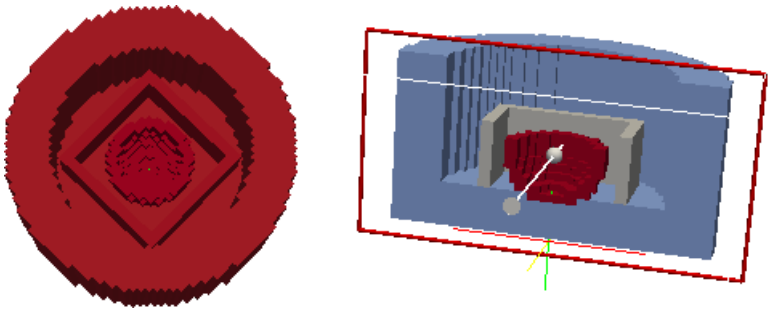

Figure 44 Different views of walls represented in fluid mesh.

Here, the wall grid of all compounds has been described in rectangular coordinates, i.e. the data field CoordSys had the value rectangular. Now we

consider the previous example but change the section WallGrid for Compound1 (CylindricalPool) and Compound3 (SpherePool). The value CoordSys is going to be cylindrical and spherical, respectively.

Representation of the same wall objects using the corresponding wall grid vtk-files is shown in figure 45.

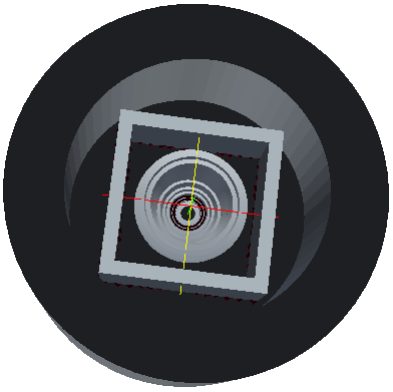

Figure 45 Wall objects represented using the corresponding wall grid vtk-files.

Once more, it can be seen that using the appropriate coordinate systems yield a geometry that is a lot smoother.

The complete version of this example is located in the folder Examples\CoPrep and is called CylindricalPool_wallGrid\Geometry.xml.

.

8 Classification of rooms

Once all compounds have been constructed inside the fluid grid, the user has to analyse if the geometry contains subrooms and if they are classified correctly. This is done with the aid of ParaView and/or the Les-files created by CoPrep, the user has to verify the correctness of the classification. This will be explained by means of the example PoolWithWall.

8.1 Link layers

It is important to find out which sub-domains (sub-rooms) are connected with each other once the water has reached a certain level. Each room that can be filled with water has to be split in separate sub-rooms if several free boundaries (at different water levels) appear (cf. figure 46).

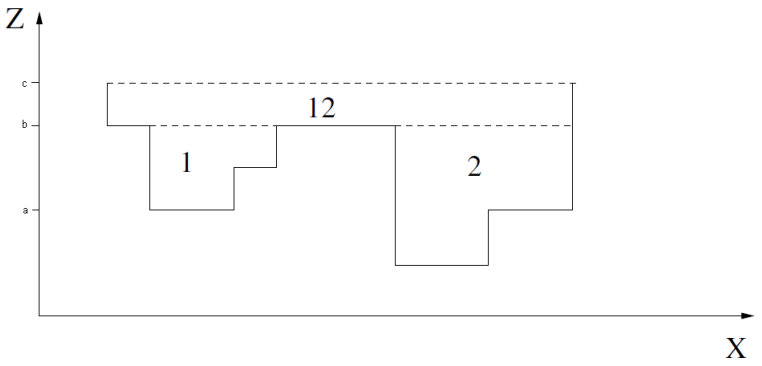

Figure 46 A vertical cut of the domain. Two sub-rooms and their common layer (which is another sub-room) can be seen.

In figure 46, one can see two sub-rooms (1 and 2) and the link layer that appears with rising water level. If the water level is below the height 'b' it is necessary to simulate the flow in two separate sub-domains (1 and 2); if the water level is higher than 'b' a third sub-room (here called 12) is indicated by the pre-processor as a link layer. Information about all link layers is given in the file "SubDomainFusion.xml".

8.2 Identification of neighboring sub-rooms

Let us consider the example PoolWithWall which describes a square pool separated by a small wall. In the figure 47 the PoolWithWall's geometry is shown.

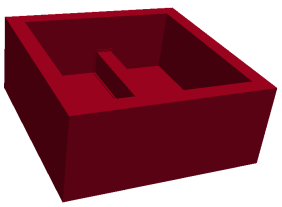

Figure 47 PoolWithWall

On the basis of this example the identification of sub-rooms will be explained. Also, it will be illustrated how to verify the correctness of the colors assigned

by CoPrep.
First of all, the file Geometry.xml will regarded (see figure 48). It can be found in the folder ~/CoPrep/PoolWithWall.

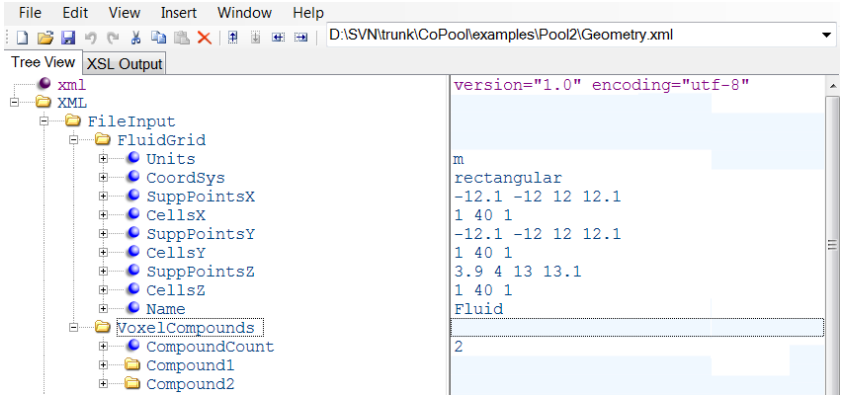

Figure 48 PoolWithWall/Geometry.xml

Geometry.xml contains two compounds. The first compound describes a square pool (it can be found in the figure 49).

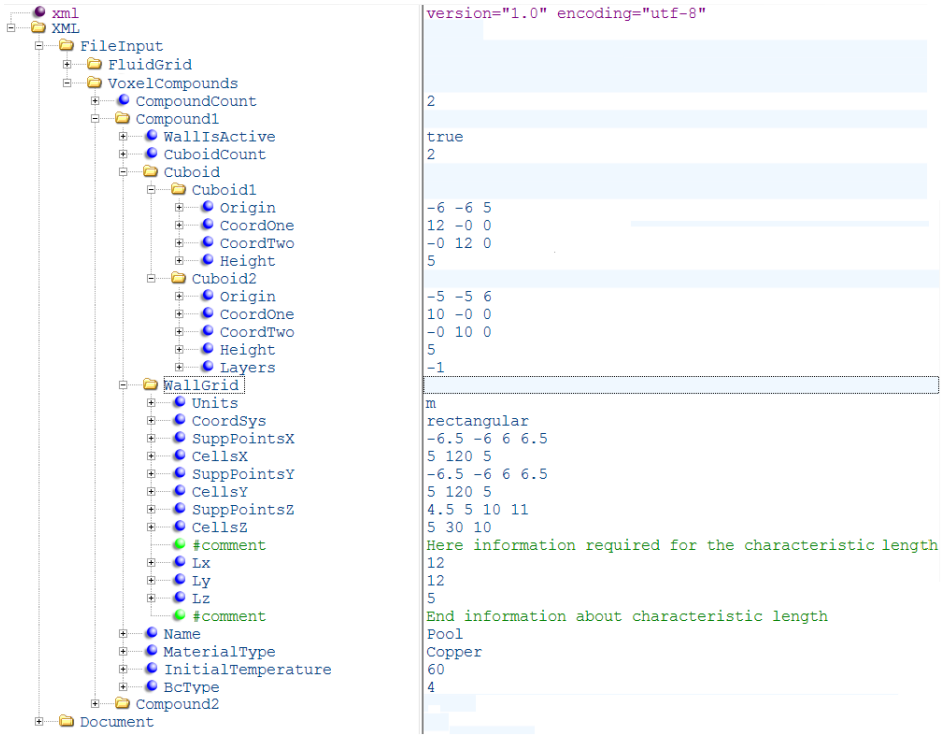

Figure 49 Parameters for Compound1: Pool

The second compound describes the wall separating the pool (see figure 50).

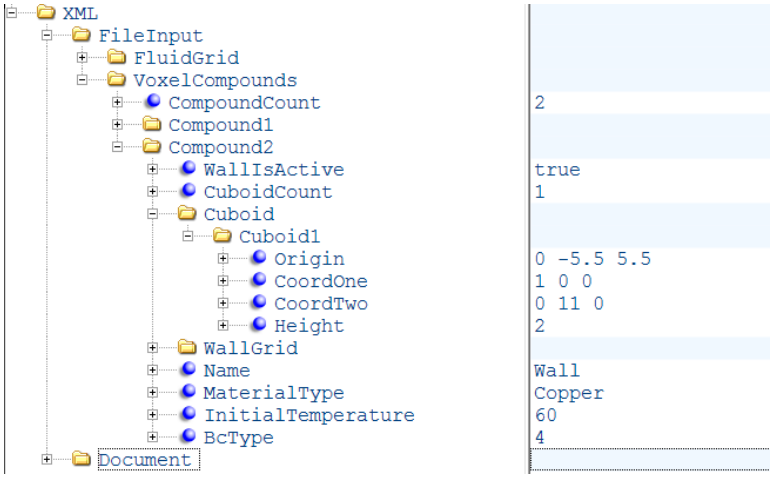

Figure 50 Parameters for Compound2: Wall

The square pool seen in figure 47 is included in the bounding box that can be filled with liquid, too. Therefore, in total, there are five different fluid colors in this geometry: around the square pool, inside the square pool on the left side of the wall, inside the square pool on the right side of the wall, inside the square pool over the wall and over the square pool. Since the fluid color count starts at two, the user should find the colors 2, 3, 4, 5 and 6.

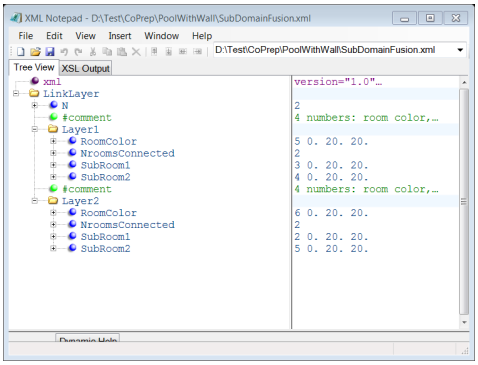

Figure 51 The content of the SubDomainFusion.xml.

The result of sub-room classification can be found in SubDomainFusion.xml file (see figure 51). In this case we have two link layers (sub-rooms with colors 5 and 6). The first one connects sub-rooms with colors 3 and 4. The second one connects the sub-rooms with colors 2 and 5. In the figure 52 different link layers are shown.

Now, it will be explained how to check that the five colors that should exist

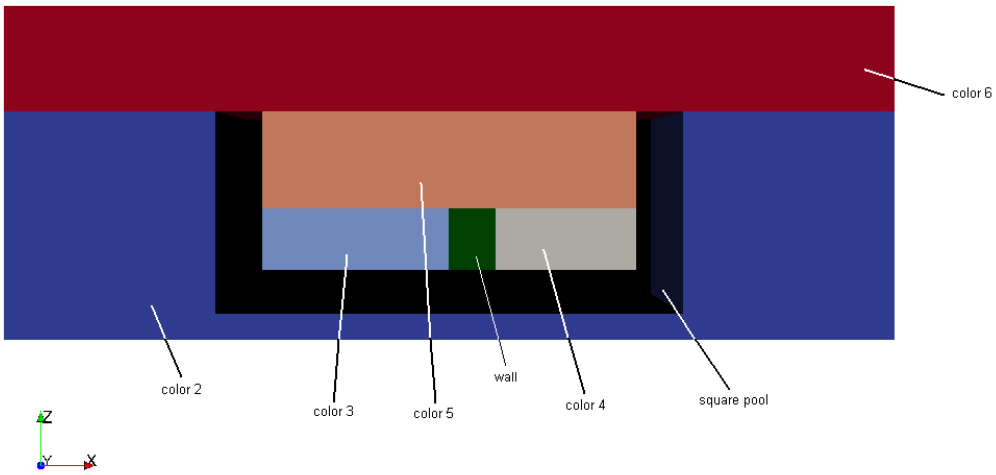

Figure 52 The colors found in the geometry of "PoolWithWall".

really do exist and that they are arranged correctly. For this, the user needs a program to visualize the results created by CoPrep, for example VisIt or ParaView. In this tutorial, the verification is explained with the help of ParaView.

There are two main steps to verify the correctness of the assigned colors:
(1) via the vtk-file of the fluid grid
(2) via the vtk-files of the compounds

After running CoPrep, there should be a folder COPREP_OUTPUT containing - amongst others - the files Fluid.vtk, Pool.vtk and Wall.vtk.

First, open the file Fluid.vtk in ParaView (if the fluid grid has another name in the geometry file, the vtk-file will have another name, too).
The user should see the bounding box of the fluid grid. Select the operation Threshold as described in section 5.1. Choose the Lower Threshold equal to two and the Upper Threshold as six (since these are the minimal and maximal expected fluid colors).

In the Object Inspector, choose the tab Display and in the block Color, click on Edit Color Map (see figure 53).

A new window opens up. Uncheck the check box "Automatically Rescale to Fit Data Range" in case it is checked and click on "Rescale Range". Choose the minimum as two and the maximum as six, then click "Rescale" and close the Color Scale Editor (see figure 54).

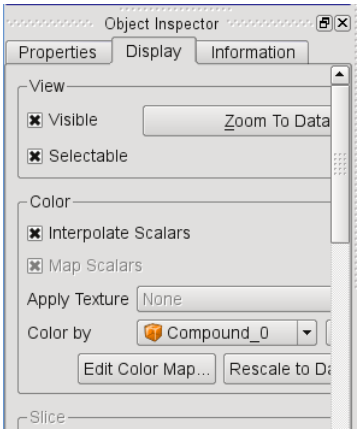

Figure 53 Object Inspector.

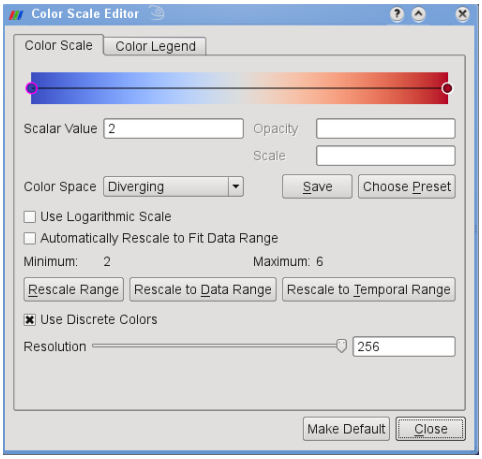

Figure 54 Color Scale Editor.

As a last step, select the operation Clip. Best visualization results are obtained by choosing the Y Normal. Now the user should see a similar result to figure 52. To be sure which domain belongs to which color number, one can change the lower und upper threshold both to two, click Apply and see the domain belonging to color two. Change the lower und upper threshold both to three, click Apply and so on.

This is a first test to see if the bounding box has been split into sub-rooms correctly. The second possibility is to take a closer look at the compounds, i.e. in this example Pool (Compound1) and Wall (Compound2). Here it shows that it is very important to add extra cells around each compound to obtain a boundary layer to save information concerning neighbors.

First of all, delete everything in the ParaView Pipeline Browser and open the

file Pool.vtk. Select Threshold and set the lower and upper threshold to two and six, respectively. Normally, the color scale is still rescaled from two to six and therefore, the image similar to image in the figure 55 should be displayed.

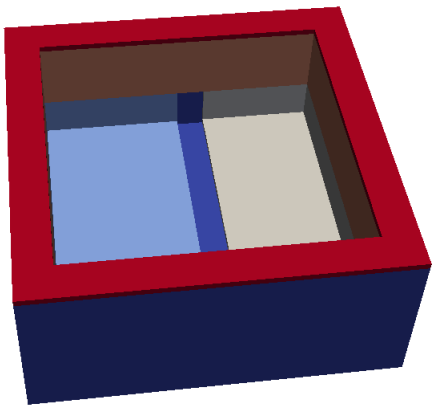

Figure 55 Pool.vtk

If the different sub-room colors cannot be differentiated, the user might need to redo the setting of the color scale as explained above (see figure 53).
Next, open the file Wall.vtk and select a threshold from two to six for it.
For both compounds, choose clip with setting Y Normal. The result similar to result in the figure 56 should be obtained.

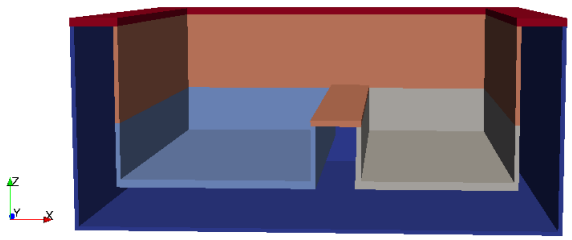

Figure 56 Result.

Remark: The "walls" one can see in figure 55 and 56 are actually no real walls but only the boundary layer consisting of the extra cells created for each compound.

9 Overlapping part

In the case of overlap of two wall objects, when a boundary cells belonging to one object, is completely inside another object, it is marked with appropriate color, which is larger or equal to 201. The overlapping part is explained here on the base of example Pool2.
In the figure 57 we show the information CoPrep prints after running example Pool2.

```
WALL MATRIX :
                                    Name   WC CS        MT   IT  BCC    CL
                                    Wall    0  0Uninitialized    0    4 201 3 112 4
                                    Pool    0  0Uninitialized    0    6 2 111 5 3 201 4


        BCC= Boundary Colour Count
        CS= Coordinate System (0=RECTANGULAR, 1=CYLINDRICAL, 2=SPHERICAL)
        MT= MaterialType
        IT= Initial Temperature
        WC= Wall Color
        CL= Color List

OVERLAP MATRIX :
                          Wall                          Pool    201
```

Figure 57 CoPrep output.

The information concerning the wall matrix should not surprise the user since it has been treated in section 5.1. New in this example is the overlap matrix. CoPrep indicates that the compound Wall overlaps with the compound Pool and that the overlap has the color 201.

Already during the construction, the user has to pay attention that two compounds have to overlap if heat exchange is supposed to take place.

By regarding the details of the xml-file displayed in figure 49 and 50 the user can see that the Pool and the Wall overlap by 0.5m.
This overlap can be demonstrated with the help of ParaView. For this, open the files Pool.vtk and Wall.vtk. For both compounds, choose the operation threshold and set the lower as well as the upper threshold to 201. Remember that 201 is the overlap color indicated by CoPrep (see figure 57) and therefore, only the overlapping parts of the compounds should be displayed. The result should looks similar to result in the figure 58.

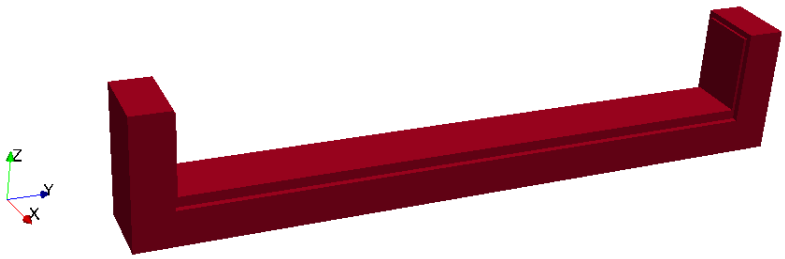

Figure 58 Overlapping parts of Pool and Wall.

10 Concluding remarks

The pre-processor is a very essential part of the CoPool software. It allows to generate geometry and meshes for simulations of liquid transport and heat conductivity in the liquid and in walls. The wall objects can be discretized in different coordinate systems. This allows to achieve the best representation of the object and solve the heat conductivity equation with high accuracy on grids with relatively small number of grid cells. The pre-processor data management allows to adapt the mesh at appropriate positions if needed. The full work flow for simulations with CoPool is described in [1].

References

[1] A. Zemitis, O.Iliev, K. Steiner and T. Gornak, CoPool, User's Manual, version 2.5.0 , Fraunhofer ITWM, Kaiserslautern, 2012.